Announcement for an upcoming Generation of TP-based educational math assistants

Comments on *eduTPS*: “Theorem-Prover based Systems for Education”

Walther Neuper

Institute for Softwaretechnology
Graz University of Technology

Working group *eduTPS* at CADGME’12
June 22 - 24, 2012
Novi Sad, Serbia

This is **NOT** the topic . . .

Outline

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	CAD/CAM DGS
applied sciences	numerical computation	Spreadsheets (SSH)
math. education	symbolic computation	
	Computer Algebra Macsyma 1968	CAS
	Theorem Proving Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
	geometry	CAD/CAM DGS
mathematics		
	numerical computation	Spreadsheets (SSH)
applied sciences		
	symbolic computation	
math. education		
	Computer Algebra Macsyma 1968	CAS
	Theorem Proving Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
	geometry	CAD/CAM DGS
mathematics		
	numerical computation	Spreadsheets (SSH)
applied sciences		
	symbolic computation	
math. education		
	Computer Algebra Macsyma 1968	CAS
	Theorem Proving Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	CAD/CAM DGS
applied sciences	**numerical computation**	Spreadsheets (SSH)
math. education	**symbolic computation**	
	Computer Algebra Macsyma 1968	CAS
	Theorem Proving (TP) Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
	geometry	CAD/CAM DGS
mathematics		
	numerical computation	Spreadsheets (SSH)
applied sciences		
	symbolic computation	
math. education		
	Computer Algebra Macsyma **1968**	CAS
	Theorem Proving (TP) Automath **1967**	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	**CAD/CAM** DGS
applied sciences	numerical computation	Spreadsheets (SSH)
math. education	symbolic computation	
	Computer Algebra Macsyma 1968	**CAS**
	Theorem Proving Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	CAD/CAM DGS
applied sciences	numerical computation	Spreadsheets (SSH)
math. education	symbolic computation	
	Computer Algebra Macsyma 1968	CAS
	Theorem Proving Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	CAD/CAM DGS
applied sciences	numerical computation	Spreadsheets (SSH)
math. education	symbolic computation	
	Computer Algebra Macsyma 1968	CAS
	Theorem Proving Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
	geometry	**CAD/CAM** DGS
mathematics		
	numerical computation	**Spreadsheets (SSH)**
applied sciences		
	symbolic computation	
math. education		
	Computer Algebra Macsyma 1968	**CAS**
	Theorem Proving Automath 1967	**TPS** **TP-based Systems**

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	CAD/CAM DGS
applied sciences	numerical computation	**Spreadsheets (SSH)**
math. education	symbolic computation	
	Computer Algebra Macsyma 1968	**CAS**
	Theorem Proving Automath 1967	TPS

Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	CAD/CAM **DGS**
applied sciences	numerical computation	**Spreadsheets (SSH)**
math. education	symbolic computation	
	Computer Algebra Macsyma 1968	**CAS**
	Theorem Proving Automath 1967	TPS

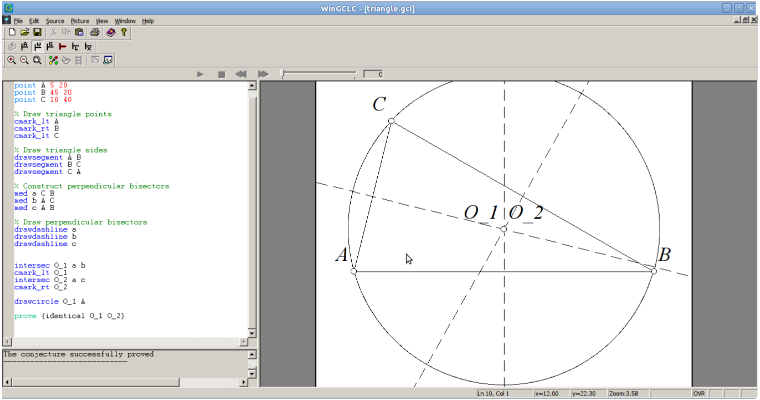
Survey on Math. Software

In science of ...	SW-tools for ...	... standardized as ..
mathematics	geometry	CAD/CAM **DGS**
applied sciences	numerical computation	**Spreadsheets (SSH)**
math. education	symbolic computation	
	Computer Algebra Macsyma 1968	**CAS**
	Theorem Proving Automath 1967	**TPS !?!**

Outline

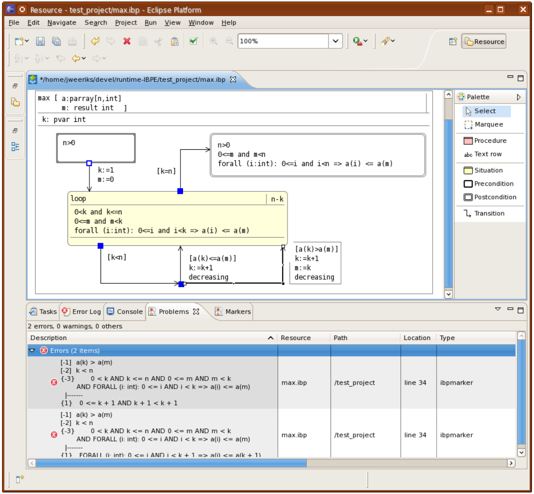
1. Survey on Mathematical Software
   - Three Examples for TP-based Math Assistants
2. Characteristics for a “New Generation”
   - Conceptual Foundations: Integration of Logics
   - Technological Features: Transparency, Flexibility
   - Expected Impact: Education, Research, Development
3. Conclusion — Invitation

GCLC, Univ. of Belgrade



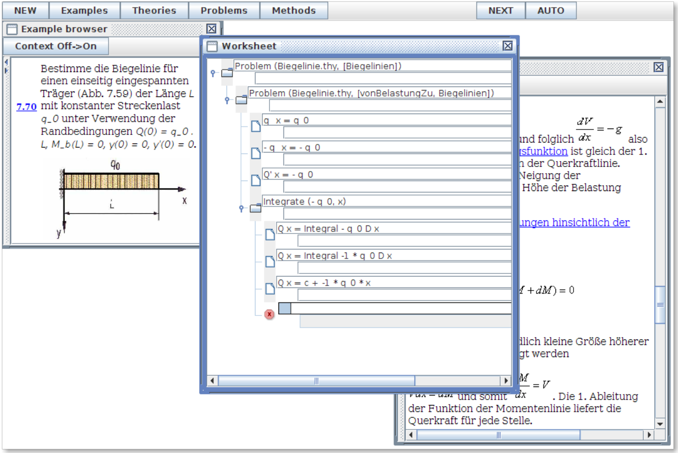
GeoGebra's "academic relative", see *prove {identical O_1 O_2}* at bottom left.

Socos, Abo Akademi Turku



For software “correct by construction” in education.

$\mathcal{ISAC}$, TU Graz & RISC Linz



For step-wise problem solving in applied mathematics.

Outline

Conceptual Foundations

Mathematics is the science of *reasoning* ...

- each operation can be *proved*
- ...to *"prove"* is the essence which distinguishes math.

TP ((Computer) Theorem *Proving*) realises this essence.

Consequences for TP-based software:

1. TP provides a logical framework for CAS, DGS, ...
2. TP is **integrative** (rather than competitive)
3. TP covers an essential range of mathematics including STEM [1].

[1] STEM: "Science, Technology, Engineering and Mathematics"

Conceptual Foundations

Mathematics is the science of *reasoning* ...

- each operation can be *proved*
- ... to *"prove"* is the essence which distinguishes math.

TP ((Computer) Theorem *Proving*) realises this essence.

Consequences for TP-based software:

1. TP provides a logical framework for CAS, DGS, ...
2. TP is **integrative** (rather than competitive)
3. TP covers an essential range of mathematics including STEM [1].

[1] STEM: "Science, Technology, Engineering and Mathematics"

Conceptual Foundations

Mathematics is the science of *reasoning* ...

- each operation can be *proved*
- ...to *"prove"* is the essence which distinguishes math.

TP ((Computer) Theorem *Proving*) realises this essence.

Consequences for TP-based software:

1. TP provides a logical framework for CAS, DGS, ...
2. TP is **integrative** (rather than competitive)
3. TP covers an essential range of mathematics including STEM [1].

[1] STEM: "Science, Technology, Engineering and Mathematics"

7 foundamental capabilities

in PISA's competence model for mathematics:

1. **Communication**: *"... perceiving the existence of some challenge and recognizing a problem situation ..."*
2. **Mathematising**: *"... transforming a problem defined in the real world to a strictly mathematical form ..."*
3. **Representation**: *"... selecting, interpreting and using a variety of representations to capture a situation ..."*
4. **Reasoning** and argument: *"... logically rooted thought processes that check a justification that is given, ..."*
5. Devising **strategies** for solving problems: *"... critical control processes that solve problems ..."*
6. Using **symbolic**, formal and technical language and **operations**: *"... within a mathematical context ..."*
7. Using mathematical **tools**: *"... being able to make use of various tools that may assist math activity ..."*

7 foundamental capabilities

in PISA's competence model for mathematics:

1. **Communication**: *"…perceiving the existence of some challenge and recognizing a problem situation …"*
2. **Mathematising**: *"…transforming a problem defined in the real world to a strictly mathematical form …"*
3. **Representation**: *"…selecting, interpreting and using a variety of representations to capture a situation …"*
4. **Reasoning** and argument: *"…logically rooted thought processes that check a justification that is given, …"*
5. Devising **strategies** for solving problems: *"…critical control processes that solve problems …"*
6. Using **symbolic**, formal and technical language and **operations**: *"…within a mathematical context …"*
7. Using mathematical **tools**: *"…being able to make use of various tools that may assist math activity …"*

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematatising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

7 foundamental capabilities . . .

. . . respectively **covered by TP-based systems**:

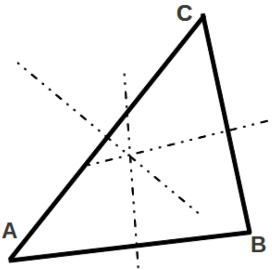
1. Communication: not *specifically* addressed by TP
2. Mathematising: **formalisation** is a prerequisite for TPS support — can be prepared and hidden from student
3. Representation: various **specifications** can be offered — and tried out using next-step-guidance
4. Reasoning: every operation in TPS has a **mechanized justification** — can be hidden and handled on request
5. Strategies: various solving **algorithms** can be offered — and tried out using next-step-guidance
6. Symbolic operations: **all** TPS operations have a **symbolic** representation — next-step-guidance helps.
7. Tools: TPS address the other capabilities above

Doesn't all that overstrain students ? Not necessarily !

For example “Mathematising”

The perpendicular midlines of the sides in a triangle meet in one point.

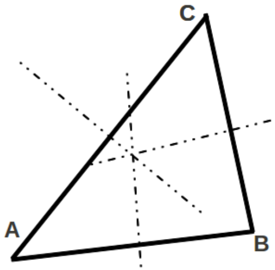
We know !



For example “Mathematising”

The perpendicular midlines of the sides in a triangle meet in one point.

We know !



For example “Mathematisising”

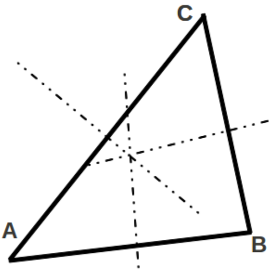
The perpendicular midlines of the sides in a triangle meet in one point.

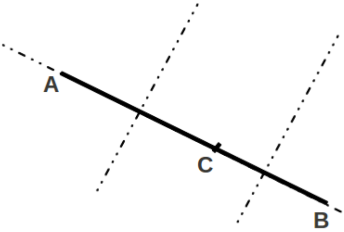
We know !

Really?

prove {identical O_1 O_2} requires **“non-degeneracy conditions”**

Without “mathematising” (specifying formally) GCLC can**not** *prove*!





For example “Mathematising”

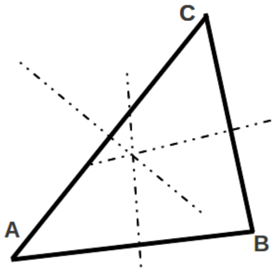
The perpendicular midlines of the sides in a triangle meet in one point.

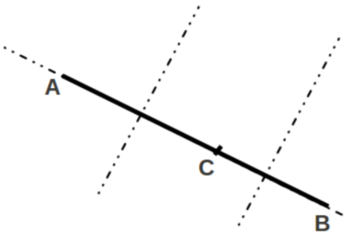
We know !

Really?

prove {identical O_1 O_2} requires **“non-degeneracy conditions”**

Without “mathematising” (specifying formally) GCLC can**not** *prove*!





For example “Mathematising”

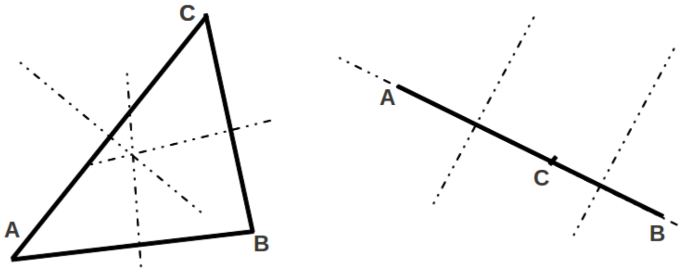
The perpendicular midlines of the sides in a triangle meet in one point.

We know !

Really?

prove {identical O_1 O_2} requires **“non-degeneracy conditions”**

Without “mathematising” (specifying formally) GCLC can**not** *prove*!



Outline

Technological Features

These features arise from TP ...

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the “LCF-paradigm” (not a program code!)
3. propose a next step if learners get stuck:
   “next-step-guidance” due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- ...

Technological Features

These features arise from TP . . .

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the "LCF-paradigm" (not a program code!)
3. propose a next step if learners get stuck:
   "next-step-guidance" due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- . . .

Technological Features

These features arise from TP . . .

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the “LCF-paradigm” (not a program code!)
3. propose a next step if learners get stuck:
   “next-step-guidance” due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- . . .

Technological Features

These features arise from TP . . .

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the "LCF-paradigm" (not a program code!)
3. propose a next step if learners get stuck:
   "next-step-guidance" due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- . . .

Technological Features

These features arise from TP . . .

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the "LCF-paradigm" (not a program code!)
3. propose a next step if learners get stuck:
   "next-step-guidance" due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- . . .

Technological Features

These features arise from TP . . .

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the “LCF-paradigm” (not a program code!)
3. propose a next step if learners get stuck:
   “next-step-guidance” due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- . . .

Technological Features

These features arise from TP ...

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the “LCF-paradigm” (not a program code!)
3. propose a next step if learners get stuck:
   “next-step-guidance” due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- ...

Technological Features

These features arise from TP ...

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the "LCF-paradigm" (not a program code!)
3. propose a next step if learners get stuck:
   "next-step-guidance" due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- ...

Technological Features

These features arise from TP . . .

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the "LCF-paradigm" (not a program code!)
3. propose a next step if learners get stuck:
   "next-step-guidance" due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- . . .

Technological Features

These features arise from TP ...

1. check user input automatically, **flexibly** and reliably:
   Input establishes a *proof situation* (for *automated* proving) with respect to the logical context
2. give explanations on request by learners:
   All underlying mathematics knowledge is **transparent** due to the “LCF-paradigm” (not a program code!)
3. propose a next step if learners get stuck:
   “next-step-guidance” due to Lucas-Interpretation.

Thus featuring software support for:

- **step-wise solving math** problems in STEM
- learning interactively like with a **chess-program**
- ...

Outline

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *"next-step-guidance"*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *“next-step-guidance”*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *“next-step-guidance”*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *"next-step-guidance"*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *“next-step-guidance”*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *“next-step-guidance”*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *“next-step-guidance”*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Impact expected . . .

- for **learners** in STEM from high-school to university:
  - *independent learning* in all phases of problem solving
  - construction of solutions with *“next-step-guidance”*
  - *flexible* access to knowledge in context of steps
- for **researchers** in science education,cognitive science
  - *logging of steps* for analysis of problem solving behav.
  - summative *assessment* of step-wise problem solving
  - mechanised *analysis of prerequisites* in curricula
- for **educational planners** and administrators
  - curriculum development on *mechanised knowledge*
  - cross-*institutional interfaces* are explicit
  - *summative assessment* of institutions
- for **developers** of systems, knowledge and dialogs
  - TP components contribute *support by automation* in checking user input, access to knowledge, . . .
  - *but* require adherence to TP standards and logics

Outline

Conclusion

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
========================
Mechanised Foundation of Math

Renewed Pedagogy for the Future
==========================
△
STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future
============================
△

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
========================
Mechanised Foundation of Math

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future

========================

△

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP

▽

========================

Mechanised Foundation of Math

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future
=========================
△
STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
=========================
Mechanised Foundation of Math

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future

==========================

△

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP

▽

========================

Mechanised Foundation of Math

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future

==========================

△

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP

▽

========================

Mechanised Foundation of Math

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
========================
Mechanised Foundation of Math

Renewed Pedagogy for the Future
===========================
△
STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future

==========================

△

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP

▽

========================

Mechanised Foundation of Math

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future
==============================
△

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
=========================
Mechanised Foundation of Math

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future

==============================

△

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience

Effective Practice in Math Education

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP

▽

=========================

Mechanised Foundation of Math

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future

==========================

△

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP

▽

========================

Mechanised Foundation of Math

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future

==========================

△

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP

▽

======================

Mechanised Foundation of Math

STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future
==============================
△
STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
=========================
Mechanised Foundation of Math

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future
============================
△
STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
========================
Mechanised Foundation of Math

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

Conclusion

Renewed Pedagogy for the Future
================================
△
STEM
individualise,
meaningful math
inquiry-based learning
embodyment, social experience
Effective Practice in Math Education

infinite variety of human thought
distilled to science of math
abstracted to logics
implemented in
software
TP
▽
===========================
Mechanised Foundation of Math

Let's hope for joint inspiration how to relate the sides !

Thank you for listening to this invitation !

TP Technology I

P. Quaresma and R.-J. Back, editors,
Proceedings First Workshop on *TP Components for Educational Software* .
Electronic Proceedings in Theoretical Computer Science, Vol. 79, 2012.

Theorem Prover Isabelle's theories,
http://isabelle.in.tum.de/dist/library/HOL/index.html

Theorem Prover Coq,
http://coq.inria.fr/